A MANY SORTED ALGEBRA BASED SIGNAL PROCESSING DATA FLOW LANGUAGE

C. R. Giardina

Stevens Institute of Technology
Hoboken, New Jersey 07030

"Abstract."

A many sorted algebra is developed for digital signal processing. It provides a rigorous setting for specifying various discrete signal processing operations. The algebra is the framework for a digital processing data flow language.

"Keywords."

Signal algebra; Data flow language; Signal flow language.

INTRODUCTION

Languages for data flow architecture are primarily directed graph based [1], and mainly involve arrows. A natural foundation for these languages is algebra where principal concepts are defined in terms of mappings using arrows [2]. As an instance of the use of the arrow, consider the notation: F: X → Y. These symbols denote the fact that the function F maps elements in the set X into elements in the set Y. In the notation involving arrows, functions are treated as entities in themselves rather than being viewed in an element by element fashion.

Data flow languages for signal processing employ opcodes whose operands are signals which are perceived as entities in themselves. Element by element representation and specification is rarely the case. This is true in languages for most concurrent architectures.

In this paper a many sorted algebra is given for signal processing applications. This algebra provides the framework for a signal processing data flow language, and is described herewithin. The concept of a many sorted algebra [3] is similar to that of a heterogeneous algebra [4] and is employed in numerous computer science applications [5-7].

A many sorted algebra involves several sets of sorts or types. Among the types of entities employed in this paper are integers, reals, discrete signals (defined in section 3), and the star extension of the reals discussed below and defined in the next section. An additional ingredient in a many sorted algebra is operators. These operators map elements in sets of various sorts into elements of a given sort. The side conditions which the operators in the algebra possibly obey, such as, commutative laws, associative laws, etc. constitutes the variety of the algebra. This concept will be illustrated in section 6.

A vector space is elegantly modeled using a many sorted algebra. There are two sorts of sets employed in that application, namely, sets of scalars and sets of vectors. In addition, there are numerous operators in a vector space. In particular, there is the binary operation of scalar multiplication. In scalar multiplication a vector is multiplied by a scalar and the output is another vector.

An algebra for digital signal processing is somewhat more complicated then that for a vector space. In a vector space fundamental operators are rigorously defined, and from these, under function composition, various other transforms are formed. In digital signal processing useful operations are given in the literature, however, many of these operations are not rigorously specified. An algebraic structure is constructed to efficiently and rigorously describe these operations. Results are only provided for the one dimensional case in this document.

The domain for signals used in one dimensional digital signal processing is K, where K is a subset of the integers Z, K⊂Z. In a single application numerous sets K could be involved. The codomain for these signals is the complex field, however, most often and in this paper the reals are used. It is not uncommon to operate on two or more signals defined on different domains. Furthermore, results of the operation is often required on a domain larger than the domain determined by the intersection of the domains of the original signals. A prime example of this is the convolution g of f and h. The standard formula for obtaining g is

$$g_i = \sum_{n=-\infty}^{\infty} h_{i-n} f_n$$

Often in applications, due to sensor limitation or for other reasons, at least one of the signals, h or f is only defined on a finite set K⊂Z. Rigorously, for these conditions the convolution can not be performed and g does not exist. However, in practice, the convolution is often performed with undefined values being interpreted as zero.

In signal conditioning applications a signal f, f: Z → R is sometimes multiplied by g, g: K → R with K finite. Rigorously, f·g is obtained which is only defined on K. In practice, the result expected is the function f unaltered, with the exception that on K, f is multiplied by g. In this case, undefined values were interpreted as being equal to the number one.

A similar situation holds in digital image processing. Part of an image f is compared with an image g. The maximum pointwise grey value of g and part of f is often desired. In practice, the resulting image would be the same as f (unaltered) on the domain of f minus the domain of g, and the maximum grey values do appear on the part of f which is compared with g. In this case, undefined acts like the "minimum grey value". The role of undefined keeps changing!

In the present paper signals are defined on all of Z. The values allowed for these signals are elements of the set S which involve real numbers or a star (*). Thus, S=R {*} ; the * should be interpreted as a value of the signal, and S is called the star extension of the reals. The * might denote a value of a signal which was not observed by a sensor. Hence, the signal is defined everywhere, but it was not observed at starred values.

The star might also arise by questioning the validity of or performing an operation on a real value of a signal. In the former case, there might be a possibility distribution [8] or believability factor associated with real values of a signal. If the believability of the respective real value is small the value becomes a *. In the next section it will also be seen that 1/0 = *.

The star may have numerous interpretations. Extensions of the reals involving more than one * are also important but will not be discussed in this paper.

Hueristic processing techniques are easily and rigorously specified by employing the many sorted algebra involving the star extension of the reals; this will be seen in section 5. Extended operations on S are defined in the next section and provide a true extension of the reals. Namely if the defined operations are restricted to the reals the "usual" operations on the reals result.

OPERATIONS IN THE STAR EXTENSION OF THE REALS

Four binary operations are defined (totally) on S using the tables below. In these tables reals are denoted by r_1 and r_2. The operations in this star extension are: addition (+); multiplication (·); maximum (∨); and minimum (∧).

+	r_2	*
r_1	r_1+r_2	*
*	*	*

·	r_2	*
r_1	$r_1 \cdot r_2$	*
*	*	*

	r_2	*
r_1	$r_1 \vee r_2$	*
*	*	*

	r_2	*
r_1	$r_1 \wedge r_2$	*
*	*	*

Two unary operations are defined on S using the tables below. The first table is for the operation of subtraction (-), s denotes any element of S, and r represents any real number. The next table is for the division (÷) operation, and r denotes any non zero real number.

s	-s
r	-r
*	*

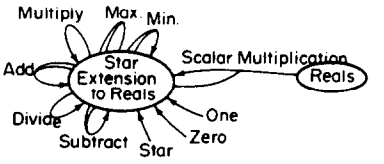
s	÷s
0	*
r	1/r
*	*

Three zero-ary operators on S, or special elements of S, are the star (*), the number one (1), and the number zero (0).

It is of value to define another type of multiplication. This multiplication (x) is two typed, very similar to vector multiplication by a scalar; for this reason it is called scalar multiplication. Specifically x: SxR → S and if r_1, r_2 denote any two real numbers then the operation becomes defined using the following table.

x	r_2
r_1	$r_1 \cdot r_2$
*	*

The polyadic (many tailed) graph [6] given below highlights the sort of elements (enclosed in the ovals) employed in the above discussion. In this diagram the names of the operators also appear along with the sort of inputs coming from the tails of the arrows; the output type is pointed to by the tip of the arrow.



From this polyadic graph scalar multiplication has two sorts of inputs; a real and a star extension of the reals and returns as an output an element in the star extension of the reals.

In the following sections the operators defined in the star extension will induce corresponding operations on discrete signals. The set of discrete signals utilized in this paper is defined in the next section along with criteria for operator inducement.

DISCRETE SIGNALS AND INDUCED OPERATIONS

A discrete signal, or one dimensional discrete signal, f is an element of S^Z, that is, it is a function from Z (the integers) into S. In this case Z is often interpreted as time. An $n \geq 1$ dimensional discrete signal is an element of S^{Z^n}, where $Z^n = ZxZx...xZ$ (the set of lattice points in n dimensions). Results are given only for the one dimensional case in this document.

Various operations on discrete signals are range induced or domain induced. These operations always have a discrete signal as one of its inputs and they output a discrete signal. The definition of range inducement will be given in a general setting since this concept is useful in n dimensional signal analysis. The operator R is said to be range induced by r, has the meaning: r is an n+k ary function $n \geq 1$, $k \geq 0$ where r:

$B^n x B_1 x B_2 x ... x B_K \to B$, and R: $(B^A)^n x B_1 x ... x B_k \to B^A$

where for any f_i in B^A i=1,2,...,n and b_j in B_j j=1,2,...,k it follows that

$R(f_1,f_2,...,f_n,b_1,b_2,...,b_k)(x) = r(f_1(x),f_2(x),...,f_n(x),b_1,b_2,...,b_k)$.

This definition will be employed with r denoting a non zero-ary operation in the star extension S. As a consequence, in this application the following sets should be equated: B=S and A=Z. Moreover, each of the seven non zero-ary operations in S shall be range induced "pointwise" operations in S^Z.

Addition will be the first illustration of a range induced operation in S^Z. For r=+, +: SxS→S the operation R=α (where α symbolizes addition) is induced where α: $S^Z x S^Z \to S^Z$. In this case, n=2, k=0 and $\alpha(f_1,f_2)(x) = f_1(x)+f_2(x)$. As a result, to add two discrete signals means to add their values pointwise. An analogous discussion occurs for r=', ∨, and ∧.

If r=÷ where ÷:S→S then let R=R (where R symbolizes reciprocal). Thus, n=1, k=0, and it follows that $R(f_1)(x) = \div f_1(x)$. The reciprocal of a dis-

crete signal is found by finding the reciprocal of each of its values pointwise.

As a final example, consider the scalar multiplication r=x, where x: SxR→S. The operation x range induces the operation R=P (P for product) where P: $S^Z x R \to S^Z$. In this case, n=1 and k=1 with B_1=R. Furthermore, $P(f_1,b_1)(x) = f_1(x) x b_1$. As a consequence to form the product of a digital signal f and a real b_1 means to multiply f(x) at each integer x by b_1. The range induced operations explained above and others based on the last section are summarized below.

Symbol Operations in S	Name of Operation	Symbol of Operation in S^Z	Range Induced Operations in S^Z Definition
+	Addition	a	$a(f_1,f_2)(x)=f_1(x)+f_2(x)$
•	Multiplication	M	$M(f_1,f_2)(x)=f_1(x)\cdot f_2(x)$
∨	Maximum (Higher)	H	$H(f_1,f_2)(x)=f_1(x)\vee f_2(x)$
∧	Minimum (Lower)	L	$L(f_1,f_2)(x)=f_1(x)\wedge f_2(x)$
-	Subtraction	S	$S(f_1)(x)=-f_1(x)$
‡	Inverse	R	$R(f_1)(x)=\frac{1}{f_1(x)}$
x	Scalar Multiplication	P	$P(f_1,b_1)(x)=f_1(x)xb_1$, b_1 real.

Somewhat similar to range inducement is the concept of domain inducement: For any n'-ary n'≥1 function d where d: $AxA_1x...xA_{n'-1}\to A$ there is an n'-ary function D such that D: $B^A xA_1x...xA_{n'-1}\to B^A$ where $D(f,a_1,a_2,...,a_{n'-1})(x)=f(d(x,a_1,...,a_{n'-1}))$. The function D is said to be domain induced from d.

Concrete examples will follow using B=S and A=Z. Only algebraic operations of relevence to digital processing shall be induced. In particular, there is a binary addition d=+ in Z which induces a translation operation T in S^Z. To see this, notice that n'=2 and +: ZxZ→Z, hence, it follows that D=T where T: $S^Z xZ \to S^Z$. By using the definition of domain inducement it follows that $T(f,a_1)(x)=f(x+a_1)$, and so the operator T translates the digital signal f,a_1 units (to the left). Another important operation in Z is unary subtraction d=- in Z. Since -: Z→Z there is an induced operator in S^Z where D=F (F stands for flip) and F: $S^Z \to S^Z$. In this case n'=1 and F(f)(x) = f(-x). A summary of these two domain induced operations is given next.

Symbol of Operation in Z	Name of Operation in Z	Symbol of Operation in S^Z	Name of Operation in S^Z	Domain Induced Operator in S^Z Definition
+	Addition	T	Translate	$T(f,a_1)(x)=f(x+a_1)$ a_1 integer
-	Subtraction	F	Flip	$F(f)(x)=f(-x)$

Several additional operations in S^Z are given in the following section.

OTHER OPERATIONS FOR DISCRETE SIGNALS

A special element of S^Z called the unity signal, denoted by 1, will be identified (as a zero-ary operator). It has "height" one at each integer, that is 1(x)=1. Two additional zero-ary operators of importance are the zero signal, denoted by 0 and the star signal, denoted by *, where 0(x)=0 and *(x)=* respectively. By using the operations from the last section it can be observed that 0=a(1,S(1)) and *=R(0).

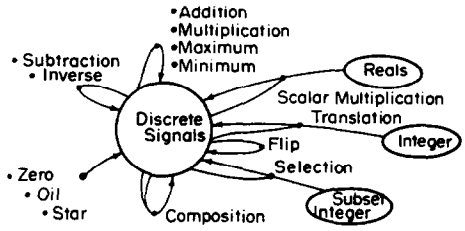
The next two operations are similar to operators found in data base algebras. The selection operator $\mathcal{S}$ is a binary operator with inputs: a discrete signal f, and a subset K of Z. The resulting discrete signal g has the same values as f on the set K, but has star values elsewhere. Thus, $\mathcal{S}$: $S^Z x 2^Z \to S^Z$, S(f,K)=g and $g(x) = \begin{cases} f(x) & x \text{ in } K \\ * & \text{elsewhere} \end{cases}$ where the set of all subsets of Z is denoted by 2^Z.

The final operator is a composition or extension operator E and is such that E: $S^Z x S^Z \to S^Z$ where $(f,g)(x) = \begin{cases} f(x) & \text{whenever } f(x)\neq * \\ g(x) & \text{elsewhere} \end{cases}$

A summary of these operators is given below.

Name of Operator	Symbol of Operation	Operator Definition
One	1	1(x)=1
Zero	0	0(x)=0
Star	*	*(x)=*
Selection	$\mathcal{S}$	$\mathcal{S}(f,K)(x) = \begin{cases} f(x) & x \text{ in } K \subseteq Z \\ * & \text{elsewhere} \end{cases}$
Composition	$\mathcal{E}$	$\mathcal{E}(f,g)(x) = \begin{cases} f(x), & f(x)\neq * \\ g(x) & \text{elsewhere} \end{cases}$

A polyadic graph is provided illustrating the sorts of sets employed and the operators rigorously defined thus far.



MACRO OPERATIONS IN DIGITAL SIGNAL PROCESSING

As memtioned in the introduction, numerous procedures in signal processing are sometimes not rigorously specified. As a result, signals and operations should be defined in a manner that will yield the desired result. When adding signals, as in convolution, undefined values are often taken as zero. This type of addition will be described in the star extension by using the operators from the last four sections. Let $f \oplus g$ denote the addition of discrete signals f and g as mentioned above. It follows that

$$(f \oplus g)(x) = \begin{cases} f(x)+g(x) & \text{whenever } f(x) \text{ and } g(x)\neq * \\ f(x) & \text{whenever } f(x)\neq * \text{ and } g(x)=* \\ g(x) & \text{whenever } g(x)\neq * \text{ and } f(x)=* \\ * & \text{elsewhere} \end{cases}$$

Theorem 1

By employing the notation previously established, the $\oplus$ type of addition is described using the star extension induced addition, and composition. Specifically: $f \oplus g = E(\, E(\, a(f,g),f)g)$.

Proof

Let Z_1,Z_2,Z_3,Z_4 be a partition of the integers Z such that Z_1 = {x: f(x) and g(x) = *} ;

Z_2 = {x: f(x) = * and g(x) ≠ *}

$Z_3 = \{x: f(x) \neq * \text{ and } g(x) = * \}$;

$Z_4 = \{x: f(x) \neq * \text{ and } g(x) \neq * \}$ then

$$a(f,g)(x) = \begin{cases} f(x)+g(x) & \text{for } x \text{ in } Z_4 \\ * & \text{for } x \text{ in } Z_1\ Z_2\ Z_3 \end{cases}$$

and

$$E(a(f,g),f) = \begin{cases} f(x)+g(x) & \text{for } x \text{ in } Z_4 \\ f(x) & \text{for } x \text{ in } Z_3 \\ * & \text{for } x \text{ in } Z_1\ Z_2 \end{cases}$$

and finally,

$$E(E(a(f,g,f),g) = \begin{cases} f(x)+g(x) & \text{for } x \text{ in } Z_4 \\ f(x) & \text{for } x \text{ in } Z_3 = f \oplus g \\ g(x) & \text{for } x \text{ in } Z_2 \\ * & \text{for } x \text{ in } Z_1 \end{cases}$$

This concludes the proof.

Macro Operators $\odot$, $\vee$ and $\wedge$ are defined similar to + namely:

$$(f \overset{\vee}{\underset{\wedge}{\odot}} g)(x) = \begin{cases} f(x) \overset{\vee}{\underset{\wedge}{\cdot}} g(x) & \text{respectively when } f(x) \text{ and } g(x) \neq * \\ f(x) & \text{whenever } f(x) \neq * \text{ and } g(x) = * \\ g(x) & \text{whenever } g(x) \neq * \text{ and } f(x) = * \\ * & \text{otherwise} \end{cases}$$

The next theorem is analogous to Theorem 1 and can be proved in the same way.

Theorem 2

$$f \odot g = E(E(M(f,g),f),g)$$
$$f \vee g = E(E(H(f,g),f),g)$$
$$f \wedge g = E(E(L(f,g),f),g)$$

Simple as well as more difficult operators exist in the discrete signal processing algebra.

Absolute value, $|\ |: S^Z \to S^Z$ is defined as $|f| = f \vee S(f) = M(f,S(f))$. It involves finding the maximum of a value and the negative of that value. The convolution operation C,

C: $S^Z \times S^Z \to S^Z$ will now be defined. So let $g = C(f,h)$ and recall from the introduction that

$$g_i = \sum_{n=-\infty}^{\infty} h_{i-n} \cdot f_n$$

$$= \cdots h_{i+1}f_{-1}+h_if_0+h_{i-1}f_1+h_{i-2}f_2+\cdots+h_{i-j}f_j+\cdots \text{ , } i \text{ in } Z.$$

Terms in the convolution involve the product of the signal f with the translation of the flip of h. The product term involving translation by n units is given by $u^i(n)$ where $u^i(n) = T(M(T(F(h),j),f)n)$. and g is the composition of all the g^i where

$$g^i = \bigoplus(\ldots u^i(-1) \oplus u^i(0) \oplus u^i(1) \oplus u^i(2) \oplus \ldots, \{i\}), -\infty < i < \infty.$$

A few comments are in order. Notice that in the discrete signal algebra "signals are defined as entities in themselves", and not in an elemental fashion. Also, in the above expression for $u^i(n)$ the multiplication M was employed, and not the multiplication $\odot$: in the motivating application undefined is understood as being equal to zero.

Macro notation such as Σ , π are defined in the algebra, but will not be discussed in this document. Also, not discussed herewithin is the convergence of the above sum, for instance, assume that f and h both have only a finite number of non star values. Finally, parenthesis are left out in the above addition because the associative law holds. Variety considerations are discussed in the next section.

ON THE VARIETY OF THE DISCRETE SIGNAL PROCESSING ALGEBRA

Various laws and identities specified by equations hold in the algebra developed herein. Many of these identities are induced by similar identities on the star extension of the reals or on the integers. The structure is seen to be a commutative monoid under both addition and multiplication. Furthermore, it is a join semilattice with greatest element under the maximum operation, and a meet semilattice with least element under the minimum operation. These properties follow from the identities satisfied by the addition, multiplication, maximum, and minimum operations given below. Addition satisfies the identities:

A1) Associative Law, $a(a(f_1,f_2),f_3)=a(f_1,a(f_2,f_3))$.

A2) Zero Law, $a(f,0) = a(0,f) = f$.

A3) Commutative Law, $a(f,g) = a(g,f)$.

The following laws hold for Multiplication:

M1) Associative Law, $M(M(f_1,f_2),f_3)=M(f_1,M(f_2,f_3))$.

M2) Identity Law, $M(f,1) = M(1,f) = f$.

M3) Commutative Law, $M(f,g) = M(g,f)$.

The Maximum (or Higher) operation satisfies the identities:

H1) Associative Law, $H(H(f_1,f_2),f_3)=H(f_1,H(f_2,f_3))$.

H2) Commutative Law, $H(f,g) = H(g,f)$.

H3) Idempotent Law, $H(f,f) = f$.

H4) Greatest Element Law, $H(f,*) = *$.

The Minimum (or Lower) operation obeys:

L1) Associative Law, $L(L(f_1,f_2),f_3)=L(f_1,L(f_2,f_3))$.

L2) Commutative Law, $L(f,g) = L(g,f)$.

L3) Idempotent Law, $L(f,f) = f$.

L4) Least Element Law, $L(f,*) = *$.

The Subtraction operation is an involution since the following is true:

S1) Involution Law, $S^2(f) = S(S(f)) = f$.

Inversion is not an involution because $R^2(0) = *$.

Scalar multiplication (or Product) satisfies properties like those in a vector space, since the following laws hold:

P1) Associative Law, $P(P(f,b_1),b_2)=P(f,b_1 \cdot b_2)$.

P2) Identity Law, $P(f,1) = f$.

P3) Distributive Law, $P(a(f,g),b)=a(P(f,b),P(g,b))$.

P4) Distributive Law, $P(f,b_1+b_2)=a(P(f,b_1),P(f,b_2))$

The first distributive law shows that multiplication by reals is distributive with respect to signal addition. The second shows that multiplication by signals is distributive with respect to the addition of reals.

All the axioms of a vector space hold for this structure with the exception of additive inverse. The translation operation satisfies the interesting, but obvious identities:

T1) Associative Law, $T(T(f,a_1),a_2)=T(f,a_1+a_2)$.

T2) Zero Law, $T(f,0) = f$.

T3) Inverse Law, $T(T(f,a_1),-a_1)=T(T(f,-a_1),a_1)=f$.

Like the subtraction operation the Flip operation is an involution since the following is true:

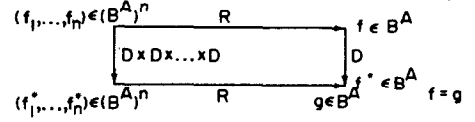
F1) Involution Law, $F^2(f) = F(F(f)) = f$.

The list of identities given here are by no means exhaustive. In particular, numerous identities such as the distributive laws or absorption laws may or may not hold between operators. Of major importance is the following theorem where the notion given in section 3 is employed.

Theorem 3

If R is a range induced operator (on B^A) and D is a domain induced operator (on B^A) then these operations commute. That is, if $R(f_1,f_2,\ldots,f_n,b_1,\ldots,b_k) = f$ and

$D(f,a_1,\ldots,a_{n-1}) = f^*$ and
$D(f_i,a_1,\ldots,a_{n'-1}) = f_i^*$, i=1,...,n' and
$R(f_1^*,f_2^*,\ldots,f_n^*,b_1,\ldots,b_k) = g$ then $f^* = g$.
Equivalently, the diagram given below commutes.



Proof

First partly traverse the diagram from the upper left hand corner by first going right and then down. Since R is range induced it follows that $f(x) = r(f_1(x),f_2(x),\ldots,f_n(x),b_1,b_2,\ldots,b_k)$. By using the fact that D is domain induced by d gives $f^*(x) = f(d(x,a_1,\ldots,a_{n-1}))$ and so

$$f^*(x) = r(f_1(d(x,a_1,\ldots,a_{n-1})),\ldots,f_n(d(x,a_1,\ldots,a_{n-1})),b_1,\ldots,b_k)$$

Next partly traverse the diagram by starting from the upper left hand corner and first going down and then to the right. From the domain inducement of D by d it follows that $f_i^*(x) = f_i(d(x),a_1,\ldots,a_{n-1})$. Finally, using the range inducement of R by r gives $g(x) = r(f_1^*(x),f_2^*(x),\ldots,f_n^*(x),b_1,\ldots,b_k)$.
The proof concludes by noticing that

$$g(x) = r(f_1(d(x,a_1,\ldots,a_{n-1})),\ldots,f_n(d(x,a_1,\ldots,a_{n-1})),b_1,\ldots,b_k)$$

and so $g(x) = f^*(x)$.

A consequence of this theorem is that translation commutes with addition, maximum, minimum, multiplication, etc. Translation corresponds to delays or advances in time in signal processing. The commutivity of translation is of crucial importance in signal processing as well as in architecture development and languages for concurrent architectures.

MANY SORTED ALGEBRA

A many sorted algebra $(S,\alpha,\Sigma,\beta,\Phi,\gamma,G)$ is a seven-tuple with components defined as follows: The first component S is a set of sorts which is always finite and nonempty. Let S* be the set of finite strings of elements from S. It is a free monoid on S. An element of S* consists of the concatenation (or sequence, without commas) of possibly repeated elements from S as well as the empty string ε.

The second component α is a function α: $S^*xS\to\Sigma$. Specifically for any two-tuple $(s_1s_2\ldots s_n,s)=v$, in S^*xS, $\alpha(v) = \Sigma_{s_1s_2\ldots s_n,s}$. Here $\Sigma_{s_1s_2\ldots s_n,s}$ is itself a set, consisting of all operator names which have domains indicated by the string of sorts $s_1s_2\ldots s_n$ and codomain of the sort s. This set is often called a signature set.

Consequently we have the third ingredient: Σ is a family of signature sets; that is the sets $\Sigma_{s_1s_2\ldots s_n,s}$ and $\Sigma_{\varepsilon,s}$ are all elements of Σ .

The fourth tuple is a function β : $S\to\Phi$ with $\beta(s)=A_s$ for every s in S, where A_s is itself a set, called the carrier set.

The fifth ingredient, Φ , is a family of carrier sets. Elements of Φ are themselves sets, namely A_s for each s in S.

The sixth component is a function γ

$$\gamma:\ \Sigma \to G,\quad (s_1s_2\ldots s_n,s)\in S^*xS \ \ s_1s_2\ldots s_n,s$$

that is γ maps the union of all signature sets into G. For any element σ in the union it follows that $\gamma(\sigma) = \sigma_A$. If σ is in $\Sigma_{s_1s_2\ldots s_n,s}$ then σ_A: $A_{s_1}xA_{s_2}x\ldots xA_{s_n}\to A_s$, that is σ is the name of the function and σ_A is the n-ary function itself.

If σ is in $\Sigma_{\varepsilon,s}$ then σ_A is in A_s. In other words a null-ary function is an element (special element) in the carrier set of the sort s.

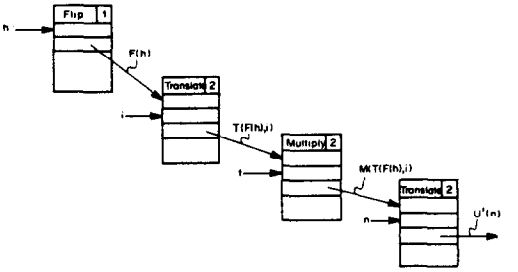
The last ingredient G is the set of all 0-ary, 1-ary, binary, trinary, etc. operators in the algebra.

The many sorted algebra for discrete signal processing shall be partially specified by identifying some of the ingredients. This is based only on the results provided in this document. As mentioned in the introduction the set of sorts involves; z=integer, r=reals, d=discrete signals, and other types which will be ignored in this development, except for s=subsets of the integers. Consequently, the set S = {z,r,d,s} shall be used. The polyadic graph given in section 4 should be referred to as the ingredients are defined.

Various non empty signature sets are: $\Sigma_{d,d}$={subtraction, inverse, flip}; $\Sigma_{\varepsilon,d}$={zero, one, star}; $\Sigma_{dd,d}$={composition, addition, multiplication, maximum, minimum}; $\Sigma_{dr,d}$={scalar multiplication} ; $\Sigma_{dz,d}$ = {translate} ; $\Sigma_{ds,d}$= {selection}. The carrier sets are $A_d=S^Z$, $A_z=Z$, $A_r=R$, and $A_s=2^Z$. The elements of G are the operators in the algebra and are obtained using γ. The function γ is such that γ(subtraction) = - where -: $S^Z\to S^Z$ defined in section 3. Also, γ(zero) = 0 defined in section 4. Furthermore, γ(selection) = S where S: $S^Zx2^Z\to S^Z$ also defined in section 4. The other operations are similarly obtained.

SIGNAL PROCESSING DATA FLOW LANGUAGE

Data flow programming graphs are easily constructed from the algebra described herein. This is best illustrated by an example. Refer to section 5 where convolution was discussed and represented using the discrete signal processing algebra. A data flow program utilizing activity templates is given to perform $u^i(n)$ in the convolution.



REFERENCES

Arbib, M.A., Manes, E.G., Arrows, Structures and Functors - The Categorical Imperative, Academic Press, N.Y., 1975.

Birkhoff, G. and Lipson, D. (1970), "Heterogeneous Algebras", Journal of Combinatorial Theory, 8, pps. 115-133.

Dennis, J.B. "Data Flow Supercomputers", Computer, Vol. 13, No. 11, Nov. 1980, pps. 48-56.

Giardina, C.R., "The Universal Imaging Algebra", Pattern Recognition Letters, Vol. 2, No. 3, March 1984, pps. 165-172.

Goguen, J.A. "et al." (1976a), "A Junction Between Computer Science and Category Theory: I, Basic Definitions and Examples," Part II, IBM Research Report RC 5908.

Goguen, J.A. (1976), "Correctness and Equivalence of Data Types", Proc. Conference on Algebraic Systems Theory, Udine, Italy. Also in Mathematical Systems Theory, pps. 352-358 (ed. G. Marchesihi and S.K. Mitter), Springer-Verlag.

Goguen, J.A. (1977), "abstract Errors for Abstract Data Types", UCLA Semantics and Theory of Computation Report 6. Also in Proc. IEIP Working Conference on Formal Description of Programming Concepts, pps. 21.1-21.32, St. Andrews, New Brunswick.

Zadeh, L.A., (1978), "Fuzzy Sets as a Basis for a Theory of Possibility", Int. J. Fuzzy Sets Syst. 1, No. 1, pps. 3-28.